NVM Express Technical Errata

Errata ID	**008**
Change Date	**5/18/2011**
Affected Spec Ver.	**NVM Express 1.0**
Corrected Spec Ver.	

Submission info

Name	Company	Date
Kevin Marks	Dell	4/6/2011
Shane Matthews	Intel	4/6/2011
Peter Onufryk	IDT	4/6/2011

This erratum clarifies that interrupt coalescing support is not required.

This erratum adds the IEEE Organization Unique Identifier and a field to indicate support for multiple ports to the Identify Controller data structure.

This erratum makes editorial changes to section 4.

Description of the specification technical flaw

Add a new paragraph to the end of section 7.5 as shown below:

The Aggregation Time field in the Interrupt Coalescing feature (refer to section 5.12.1.8) indicates the host desired maximum delay that a controller may apply to a Completion Queue entry before an interrupt is signaled to the host. This value is provided to the controller as a recommendation by the host and a controller is free to generate an interrupt before or after this aggregation time is achieved. A controller may apply this value on a per vector basis or across all vectors. The specific manner in which this value is used by the interrupt aggregation algorithm implemented by a controller is implementation specific.

Although support of the Get Features and Set Features commands associated with interrupt coalescing is required, the manner in which the Aggregation Threshold and Aggregation Time fields are used is implementation specific. For example, an implementation may ignore these fields and not implement interrupt coalescing.

Modify Figure 65 as shown below:

Figure 65: Identify – Identify Controller Data Structure

Bytes	O/M	Description
Controller Capabilities and Features		
72	M	**Recommended Arbitration Burst (RAB):** This is the recommended Arbitration Burst size. Refer to section 4.7.
75:73	M	**IEEE OUI Identifier (IEEE):** Contains the Organization Unique Identifier (OUI) for the controller vendor. The OUI shall be a valid IEEE/RAC assigned identifier that may be registered at http://standards.ieee.org/develop/regauth/oui/public.html.
76	O	**Multi-Interface Capabilities (MIC):** This field specifies whether there are multiple physical PCI Express interfaces to the host and associated capabilities. Host software can identify PCI Express interfaces that correspond to the same underlying NVM Express device by matching their unique identifiers, defined in section 7.7. Bits 7:1 are reserved. Bit 0 if set to '1' then the controller supports multiple physical PCI Express interfaces to the host. If cleared to '0' then the controller does not support multiple physical PCI Express interfaces to the host.
255:~~73~~ 77		Reserved

Modify section 4.1 as shown below:

4.1 Submission Queue & Completion Queue Definition

The Head and Tail entry pointers correspond to the Completion Queue Head Doorbells and the Submission Queue Tail Doorbells defined in section ~~3.1.10~~ 3.1.11 and ~~3.1.11~~ 3.1.10. The doorbell registers are updated by host software.

The submitter of entries to a queue uses the current Tail entry pointer to identify the next open queue entry space. The submitter increments the Tail entry pointer after submitting the new entry to the open queue entry space. If the Tail entry pointer increment exceeds the queue size, the Tail entry shall roll to zero. The submitter ~~can~~ may continue to submit entries to the queue as long as the Tail entry is less than the Current Head entry pointer. **(RETURN ADDED)**

Note: The submitter shall take queue wrap conditions into account.

The consumer of entries on a queue uses the current Head entry pointer to identify the next entry to be pulled off the queue. The consumer increments the Head entry pointer after retrieving the next entry from the queue. If the Head entry pointer increment exceeds the queue size, the Head entry pointer shall roll to zero. The consumer may continue to remove entries from the queue as long as the Head entry pointer is greater than the Tail entry pointer. **(RETURN ADDED)**

Note: The consumer shall take queue wrap conditions into account.

Creation and deletion of Submission Queue and associated Completion Queues need to be ordered correctly by host software. Host software shall create the Completion Queue before creating any associated Submission Queue. Submission Queues may be created at any time after the associated Completion Queue is created. Host software shall delete all associated Submission Queues prior to deleting a Completion Queue. Host software should only delete a Submission Queue after it has been brought to an idle condition with no outstanding commands.

If host software writes an invalid value to the Submission Queue Tail Doorbell or Completion Queue Head Doorbell register and an Asynchronous Event Request command is outstanding, then an asynchronous event is posted to the Admin Completion Queue with a status code of Invalid Doorbell Write Value. This condition may be caused by host software attempting to add an entry to a full Submission Queue or remove an entry from an empty Completion Queue. The behavior if a command is overwritten is undefined.

If there are no free completion queue entries in a Completion Queue, then the controller shall not post status to that Completion Queue until completion queue entries become available. In this case, the controller may stop processing additional Submission Queue entries associated with the affected Completion Queue until completion queue entries become available. The controller shall continue processing for other queues.

Modify the first paragraph of section 4.1.1 as shown below:

The queue is Empty when the Head entry pointer equals the Tail entry pointer. Figure 4 defines the Empty Queue condition.

Modify the first paragraph of section 4.1.2 as shown below:

The queue is Full when the Head equals one more than the Tail. The number of entries in a queue when full is one less than the queue size. Figure 5 defines the Full Queue condition. **(RETURN ADDED)**

Note: Queue wrap conditions shall be taken into account when determining whether a queue is Full.

Modify section 4.1.3 as shown below:

The Queue Size is indicated in a 16-bit 0’s based ~~parameter~~ field that indicates the number of entries in the queue. The minimum size for a queue is two entries. The maximum size for either an I/O Submission Queue or an I/O Completion Queue is defined as 64K entries, limited by the maximum queue size supported by the controller that is reported in the CAP.MQES field. The maximum size for the Admin Submission and Admin Completion Queue is defined as 4K entries. One entry in each queue is not available for use due to Head and Tail entry pointer definition.

Modify section 4.1.4 as shown below:

4.1.4 Queue ~~ID~~ Identifier

Each queue ~~can be~~ is identified through a 16-bit ID value that is assigned to the queue when it is created.

Modify section 4.1.5 as shown below:

If the weighted round robin with ~~an~~ urgent priority class arbitration mechanism is supported, then host software may assign a queue priority service class of Urgent, High, Medium or Low. If the weighted round robin with ~~an~~ urgent priority class arbitration mechanism is not supported, then ~~this field~~ the priority setting is not used and is ignored by the controller.

Modify section 4.2 as shown below:

Each command is 64 bytes in size.

Command Dword 0, Namespace Identifier, Metadata Pointer, PRP Entry 1, and PRP Entry 2 have common definitions for all Admin commands and NVM commands. Metadata Pointer, PRP Entry 1, and PRP Entry 2 ~~may~~ are not be used by all commands. Command Dword 0 is defined in Figure 6.

Figure 6: Command Dword 0

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>31:16</td><td><b>Command Identifier (CID):</b> This field ~~indicates~~ specifies a unique identifier for the command when combined with the Submission Queue identifier.</td></tr>
<tr><td>15:10</td><td>Reserved</td></tr>
<tr><td>09:08</td><td><b>Fused Operation (FUSE):</b> In a fused operation, a complex command is created by “fusing” together two simpler commands. Refer to section 6.1. This field ~~indicates~~ specifies whether this command is part of a fused operation and if so, which command it is in the sequence.
<table>
<tr><th>Bits</th><th>Definition</th></tr>
<tr><td>00b</td><td>Normal operation</td></tr>
<tr><td>01b</td><td>Fused operation, first command</td></tr>
<tr><td>10b</td><td>Fused operation, second command</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>07:00</td><td><b>Opcode (OPC):</b> This field ~~indicates~~ specifies the opcode of the command to be executed.</td></tr>
</table>

The 64 byte command format for the Admin Command Set and NVM Command Set is defined in Figure 7. Any additional I/O ~~command set~~ Command Set defined in the future may use an alternate command size or format.

Modify section 4.2 as shown below (continued):

Figure 7: Command Format – Admin and NVM Command Set

Bytes	Description
63:60	**Command Dword 15 (CDW15):** This field is command specific Dword 15.
59:56	**Command Dword 14 (CDW14):** This field is command specific Dword 14.
55:52	**Command Dword 13 (CDW13):** This field is command specific Dword 13.
51:48	**Command Dword 12 (CDW12):** This field is command specific Dword 12.
47:44	**Command Dword 11 (CDW11):** This field is command specific Dword 11.
43:40	**Command Dword 10 (CDW10):** This field is command specific Dword 10.
39:32	**PRP Entry 2 (PRP2):** This field contains the second PRP entry for the command or if ~~. If~~ the data transfer spans more than two memory pages, then this field is a PRP List pointer.
31:24	**PRP Entry 1 (PRP1):** This field contains the first PRP entry for the command or a PRP List pointer depending on the command.
23:16	**Metadata Pointer (MPTR):** This field contains the address of a contiguous physical buffer of metadata. This field is only used if metadata is not interleaved with the logical block ~~LBA~~ data, as specified in the Format NVM command. This field shall be Dword aligned.
15:08	Reserved
07:04	**Namespace Identifier (NSID):** This field ~~indicates~~ specifies the namespace that this command applies to. If the namespace is not used for the command, then this field shall be cleared to 0h. If a command shall be applied to all namespaces on the device, then this value shall be set to FFFFFFFFh.
03:00	**Command Dword 0 (CDW0):** This field is common to all commands and is defined in Figure 6.

Modify the first paragraph of section 4.3 as shown below:

A physical region page (PRP) entry is a pointer to a physical memory page. The size of the physical memory page is configured by host software in CC.MPS. Figure 8shows the layout of a PRP entry that consists of a Page Base Address and an Offset. The size of the Offset field is determined by the physical memory page size configured in CC.MPS.

Modify the last paragraph of section 4.3 as shown below:

The first PRP entry contained within the command may have a non-zero offset within the memory page. The first PRP List entry~~,~~ (i.e. the first pointer to a memory page containing additional PRP entries)~~,~~ that ~~when~~ if present is contained in the PRP Entry 2 location within the command, may also have a non-zero offset within the memory page. All other PRP and PRP List entries shall have a memory page offset of 0h, i.e. the entries are memory page aligned based on the value in CC.MPS. The last entry within a memory page, as indicated by the memory page size in the CC.MPS field, shall be a PRP List pointer if there is more than a single memory page of data to be transferred.

Modify section 4.4 as shown below:

Metadata may be supported for a namespace as either part of the logical block ~~LBA~~ (creating an extended logical block ~~LBA~~ which is a larger logical block ~~LBA~~ that is exposed to the application) or it may be transferred as a separate contiguous buffer of data. The metadata shall not be split between the logical block ~~LBA~~ and a separate metadata buffer. Refer to section 8.2.

In the case where the namespace is formatted to transfer the metadata as a separate contiguous buffer of data, then the Metadata Region is used. In this case, the location of the Metadata Region is indicated by the Metadata Pointer within the command. The Metadata Pointer within the command shall be Dword aligned.

The controller may support several physical formats of logical block ~~LBA~~ size and associated metadata size. There may be performance differences between different physical formats. This is indicated as part of the Identify Namespace data structure.

If the namespace is formatted to use end-to-end data protection, then the first eight bytes or last eight bytes of the metadata is used for protection information (specified as part of the NVM Format operation).

Modify the first paragraph of section 4.5 as shown below:

An entry in the Completion Queue is 16 bytes ~~(4 Dwords)~~ in size. Figure 11 describes the layout of the Completion Queue Entry ~~this~~ data structure. The contents of Dword 0 is command specific. If a command uses Dword 0, then the definition of this Dword is contained within the associated command definition. If a command does not use Dword 0, then the field is reserved. Dword 1 is reserved. Dword 2 is defined in Figure 12 and Dword 3 is defined in Figure 13. Any additional I/O ~~command set~~ Command Set defined in the future may use an alternate Completion Queue entry size or format.

Modify Figure 12 as shown below:

Figure 12: Completion Queue Entry: DW 2

Bit	Description
31:16	**SQ Identifier (SQID):** Indicates the Submission Queue that the associated command was issued to. This field is used by host software when more than one Submission Queue shares a single Completion Queue to uniquely determine the command completed in combination with the Command Identifier (CID).
15:00	**SQ Head Pointer (SQHD):** Indicates the current Submission Queue Head pointer for the Submission Queue indicated in the SQ Identifier field. This is used to indicate to the host Submission Queue entries that have been consumed and may be re-used for new entries. Note: The value returned is the value of the SQ Head pointer when the completion queue entry was created. By the time host software consumes the completion queue entry, the controller may have an SQ Head pointer that has advanced beyond the value indicated.

Modify the first paragraph of section 4.5.1 as shown below:

The Status Field defines the status for the command indicated in the completion queue entry, defined in Figure 14.

Modify Figure 14 as shown below:

Figure 14: Completion Queue Entry: Status Field

Bit	Description
31	**Do Not Retry (DNR):** If set to '1', indicates that if the same command is re-issued it is expected to fail. If cleared to '0', indicates that the same command may succeed if retried. If a command is aborted due to time limited error recovery (refer to section 5.12.1.5), this field should be cleared to '0'.
30	**More (M):** If set to '1', there is more status information for this command as part of the Error Information log that may be retrieved with the Get Log Page command. If cleared to '0', there is no additional status information for this command. Refer to section 5.10.1.1.
29:28	Reserved
27:25	**Status Code Type (SCT):** Indicates the status code type of the completion queue entry. This indicates the type of status the controller is returning.
24:17	**Status Code (SC):** Indicates a status code identifying any error or status information for the command indicated.

Modify section 4.5.1.1 as shown below:

Completion queue entries indicate a status code type for the type of completion being reported. Figure 15 specifies the status code type values and descriptions.

Figure 1: Status Code – Status Code Type Values

Value	Description
0h	**Generic Command Status:** Indicates that the command specified by the Command and Submission Queue identifiers in the completion queue entry has completed. These status values are generic across all command types, and include such conditions as success, opcode not supported, and invalid field.
1h	**Command Specific Errors:** Indicates an error that is specific to a particular command opcode. Errors such as invalid firmware image or exceeded maximum number of queues is reported with this type.
2h	**Media Errors:** Any media specific errors that occur in the NVM or data integrity type errors shall be of this type.
3h – 6h	Reserved
7h	**Vendor Specific**

Modify section 4.5.1.2 as shown below:

The Status Code (SC) field in the completion queue entry ~~specifies~~ indicates more detailed status information about the completion being reported.
Each Status Code set of values is split into three ranges:

- 00h – 7Fh: Applicable to Admin ~~command set~~ Command Set, or across multiple command sets.
- 80h – BFh: I/O Command Set Specific status codes.
- C0h – FFh: Vendor Specific status codes.

If there are multiple status codes that apply to a particular command failure, the controller shall report the status code with the lowest numerical value.

Disposition log

4/6/2011	Erratum captured.
4/18/2011	Added IEEE identifier registration website; converted to mandatory.
4/21/2011	Added multiple port indication to Identify Controller.
4/26/2011	Updated multiple port to multiple PCI Express physical interface.
5/18/2011	Added information on how the host can identify the device in multi-interface cases.
6/20/2011	Erratum ratified.

Technical input submitted to the NVMHCI Workgroup is subject to the terms of the NVMHCI Contributor's agreement.